Laboratorul 01.

Observație

Datele de intrare se vor da fie ca parametri în linia de comandă, fie ca valori fixe ale aplicațiilor!

Problema 1

Să se introducă programul de mai jos (clasa ***Test***) într-un fișier cu numele `Prob1.java`, folosind mediul integrat NetBeans / Eclipse.

1. Să se compileze și să se ruleze următorul program:

```
class Test {
        public static void main(String args[]) {
                System.out.println("Test Java");
        }
}
```

2. Adăugați atributul ***public*** înaintea cuvântului cheie ***class***, recompilați și executați programul. Rezolvați problema apărută!

IMPORTANT!

Clasele care **NU** au atributul ***public*** pot avea un nume diferit de numele fișierului sursă, însă dacă adăugăm atributul ***public*** este obligatoriu ca numele clasei și numele fișierului sursă, în care se află clasa, să coincidă.

Problema 2

Să se scrie o clasă ***Problema2*** care conţine două metode (funcţii):

- metoda ***print*** nestatica care primește un argument de tip `String` pe care-l afișează;
- metoda ***main*** care apelează funcția ***print()*** pentru afișarea unui șir constant.

IMPORTANT!

Metoda print se va apela în ***main*** astfel:

```
Problema2 obiect = new Problema2(); // creare obiect de tip Problema2
obiect.print("Test"); //apelare metoda print
```

Problema 3

1. Să se modifice clasa scrisă la exerciţiul anterior prin definirea a două clase, fiecare conţinând câte o metodă:
   - o clasă pentru metoda ***main***;
   - o clasă pentru metoda ***print***.
   - Să se verifice dacă ambele clase din fișier pot fi publice. Apelul metodei ***print*** se va realiza în acelaşi mod ca la problema 2!
2. Să se modifice programul anterior prin crearea a două fişiere sursă, fiecare conţinând o clasă cu câte o singură metodă. Incercati să executați ambele clase.

IMPORTANT!

Într-un fișier **NU** pot fi definite două **clase publice**.

Problema 4

Să se scrie un program pentru afișarea tuturor argumentelor primite în linia de comandă.

Observație

Argumentele se pot transmite astfel:

- daca programul se ruleaza din NetBeans / Eclipse se vor urma instructiunile de la începutul laboratorului;
- daca programul se ruleaza din linia de comandă: `java numeprogram arg1 arg2 arg3`

Problema 5

Să se realizeze o clasă care cuprinde o metodă recursivă (nestatica) care calculează puterea întreagă a unui numar întreg și o metodă pentru afișarea rezultatului funcției, alături de rezultatul funcției statice `Math.pow(baza, exp)` pentru a se putea valida. Clasa va conține un ***main*** în care se vor testa cele două metode definite anterior.

IMPORTANT!

O ***metodă statică*** a unei clase se apelează prin:

`NumeClasă.numeMetodăStatică(…)`

`Math.pow(…);`

Problema 6

Să se implementeze o clasă cu două metode:

- o metodă (nestatica) de tip `boolean` care verifică dacă un număr întreg dat este prim;
- metoda `main` care verifică metoda anterioară pentru toate numerele naturale mai mici ca 20.

Problema 7

Să se scrie un program pentru verificarea ipotezei lui Goldbach pentru primele n numere pare, prin afișarea tuturor sumelor de două numere prime prin care poate fi exprimat un număr par. Variabila n poate fi inițializată cu o valoare constantă.

Ipoteza lui Goldbach: orice nr par poate fi descompus ca sumă de cel puțin o pereche de două numere prime. Se consideră 1 ca fiind număr prim.

Pentru afișarea unei expresii de forma `a = b + c` se va scrie:

```
System.out.println(a + " = " + b + " + " + c);
```

unde `a`, `b`, `c` sunt variabile numerice de orice tip (`short`, `int`, `long`, `float`, `double`).

Problema 8

Să se scrie un program pentru ordonarea unui vector de numere și căutarea binară în acest vector, folosind metodele statice ***sort()*** și ***binarySearch()*** din clasa ***Arrays***. Vectorul va conține numere generate aleator

folosind metoda statică ***random()*** din clasa ***Math***, cu rezultat de tip ***double***.

poo/laboratoare/01.txt · Last modified: 2022/10/07 14:02 by mihai.nan